Chapter 21: Product Distribution Server Configuration

This chapter describes how to configure and manage a **UPS** product distribution node. It was written with the assumption that the reader who is setting up a distribution server has appropriate system privileges and sufficient administrative experience to create accounts, change network service configurations, and so on.

The Computing Division's primary distribution node at Fermilab is *fnkits.fnal.gov.* It is used in examples throughout this chapter, but the chapter is intended to be a general reference, and the specifics of the *fnkits* configuration are detailed in Chapter 22: *Configuration of the fnkits Product Distribution Node*.

Distribution servers like *fnkits.fnal.gov* provide a convenient central repository for product installations, but setting them up properly takes a bit of effort. An **FTP** server, a Web server, and the **UPD** configuration file (described in Chapter 32: *The UPD Configuration File*) on the server need to work together to create the right environment. This is especially important if restricted access to certain products is needed (e.g., proprietary products, or products that must be restricted to particular systems or domains only). In addition, various guidelines need to be followed in order to maintain security and keep unauthorized users from gaining control of your distribution server.

We begin the chapter by presenting step-by-step sequences of how a product distribution server responds to the two most common **UPD** commands (**`upd addproduct`** and **`upd install`**). This discussion is intended to help you understand how all the elements of a distribution server work together to execute these and other **UPD** commands. We hope that it helps put in context the material in the remainder of the chapter, which consists mostly of administration and configuration issues.

21.1 How A Server Responds to a UPD Client Command

The two commands that a distribution server receives most frequently are **`upd addproduct`** and **`upd install`**, used to add products *to* the distribution database and to download products *from* the distribution database,

respectively. Here we present step-by-step sequences of how these two processes work. As you read through the sequences of actions that follow, pay attention to which program is taking each action.

21.1.1 The Process for upd addproduct

The **`upd addproduct`** command is used to upload a product to a **UPS** product database on a distribution server. It operates by making a series of network connections to the server. All calls are made from the client system to the distribution server, who reports back results on the same data channel:

- The Web server on the distribution node is called, and a script called `ups.cgi` is used to determine if the specified product instance already exists on the distribution node. If it exists, **UPD** on the client machine prints an error and exits. If it doesn't exist, the process continues.
- The anonymous **FTP** server on the distribution node is called, and the product tar file (if any) is transferred from the user node into `/incoming`.
- The Kerberos kshd server is called, and a script called `k5arc` is used to call a script called `upd` which runs `upd move_archive_file`. This makes a product directory for the instance on the distribution node (as defined by the distribution node's `updconfig` file), installs the tar file as `${UPS_PROD_DIR}.tar` (or `${UPS_PROD_DIR}.tar.gz` or `${UPS_PROD_DIR}.zip`, according to its suffix), and unwinds part of the tar file to make the `README` file and the `ups` and `man` directories available, if present. It reports back the database, product directory, and tar file location to the client **`upd addproduct`** command.
- The anonymous **FTP** server is called, and the product's `ups` directory tar file is uploaded to `/incoming`. (If the user specified a `ups` directory, it gets uploaded over the one that was unwound from the tar file.)
- The Kerberos kshd server is called, and `k5arc` is used to call a script called `upd` which runs `upd moved_ups_dir`. This makes a `ups` directory on the distribution node for the product (as defined by the `updconfig` file) and unwinds the `ups` directory tar file. It reports back the database and `ups` directory to the client **`upd addproduct`** command.
- The **FTP** server and Web server are similarly called to install the table file.
- Finally, the Web server is called and `k5arc` runs a script called `ups` which declares the product into the distribution database.

A subset of these steps is performed to execute **`upd modproduct`** or to add a product that has a subset of these elements (e.g., one that does not include a `ups` directory).

21.1.2 The Process for upd install

The **`upd install`** command is used to download a product from a distribution node **UPS** database to a user machine. As for **`upd addproduct`**, all network connections come from the client system running **`upd install`** to the distribution server who reports any results back along that connection.

- The Web server on the distribution node is called, and `ups.cgi` is used to determine if the product instance in question exists on the server, what its dependencies are. A call to the local **UPS** determines whether the product and its dependencies exist on the user node. For the product itself and for each dependency not found on the user node, the remaining steps are taken:
- The Web server is called, and `ups.cgi` is used to determine particular details of the product on the distribution node (e.g., archive file location, product root directory, `ups` directory, and so on). If no archive file location is given, **UPD** manufactures a tar file that should work, assuming the **FTP** server can make a tar file of directories on the fly.[1] The tar file gets named according to the convention: `ftp://host/$UPS_PROD_DIR/..tar` (a "`.`" for the path, followed by "`.tar`").
- The **FTP** server is used to transfer and unwind the archive file, an archive of the `ups` directory, and the table file for the product.
- **UPD** declares the product on the local system.

21.2 Accounts Required for Distribution Server

A minimum of four separate user accounts are required for managing a distribution server.

- an account under which the Web server `ups.cgi` script will run, and which will own the products on the distribution server; usually set to *updadmin*

1. A **WU-FTP** compatible **FTP** server is used to make tar files "on the fly".

- an account under which the anonymous **FTP** server will run; usually set to *ftp*
- an account which can configure the administrative files for the Web server and the **FTP** server, a suggested name is *wwwadm* (can be any account, e.g., the maintainer's usual account)
- a *k5arc* account through which all accesses to the server are made, and which has a restricted login shell such that `ksu` is the only allowed command

Each of these accounts has particular needs and functions, and for security reasons they should be distinct from one another, as described in the following sections.

21.2.1 The updadmin Account

The *updadmin* account (which owns the cgi scripts, the k5arc and related scripts, and the products in the distribution database) has the fewest requirements. It should be usable by anyone needing to perform administrative functions related to the distribution node's **UPS** database. It should be able to schedule **cron** jobs to perform log file cleanup, reporting, and so on. It needs write access to the distribution database, the products area, and the Web server log area.

This account should *not* have write access to any of the Web server or **FTP** server configuration files.

21.2.2 The ftp Account

The *ftp* account is the home of the anonymous **FTP** service, and thus has the most restrictions on it.

The location of the *ftp* account's home directory is an important decision. The distribution node **UPS** database needs to be a subdirectory of `~ftp`**,** as do all the product roots and tar files for products that are to be distributed. Very often, then, `~ftp` is a whole separate file system.

Since this account hosts the anonymous **FTP** service, several security issues are of critical importance for setting it up securely. They are summarized here from the on-line document `http://www.cert.org/ftp/tech_tips/anonymous_ftp_config`:

- The home directory `~ftp` should not be owned by the *ftp* account. In fact, nothing whatsoever should be owned by this account. For a **UPD** server configuration, *updadmin* or perhaps *root*, would be an appropriate owner of `~ftp`.

- For the command **`upd addproduct`** to work, there must be a `~ftp/incoming` directory, writable but not readable by the *ftp* account. This directory must be readable by the *updadmin* account, however. We recommend having it owned by *updadmin* and set to mode 733.
- The anonymous **FTP** area `~ftp` needs `~ftp/etc/passwd` and `~ftp/etc/groups` files. These files should not be copies of the real system and group files. They should instead contain only the userids and groups of the files that will be encountered in the **FTP** area (`~ftp`), and should of course contain no passwords.
- The `~ftp/bin` area should contain only **ls, tar, gzip**, and **gunzip**. The `~ftp/usr/lib` area needs sufficient shared libraries to let these run. You can use **`chroot`** to test that the command runs (as in **`chroot ~ftp /bin/ls -l`** which will run the **`~ftp/bin/ls`** command under the "**chroot**ed" environment in which the **FTP** server will be living).

The **FTP** home area will be accessed via two separate avenues. The Web server will access it via its full pathname, `~ftp`, but the **FTP** server accesses this area via a **`chroot`** command. Because of these different access methods, the *ftp* account needs some symbolic links such that something chrooted to `~ftp` still finds files if the expanded `~ftp` pathname is used. For example, if `~ftp` is `/home/ftp`, then you should have symbolic links for both directory components: `home` and `ftp`. For example, when you run

```
% ls -l ~ftp
```

you should see output that contains:

```
ftp -> .
home -> .
...
```

You can create this by executing the commands:

```
% ln -s . ftp
% ln -s . home
```

This is an example of arranging things so that the **FTP** server and the Web server get a consistent view of the world, even though one uses **chroot** and the other one doesn't.

21.2.3 The wwwadm Account

This account has control of the configuration files for the **FTP** and Web servers. We refer to this account as *wwwadm* throughout this document, although this particular name is not required. Any account can be used for this,

including the regular login account of the distribution server administrator, or even *root*. Similarly, a UNIX group could be created, and people in that group could be granted access to the configuration files.

The person working under this account could seriously affect the security of the distribution server by misconfiguring either of these services, therefore we recommend that access be tightly controlled.

21.2.4 The k5arc Account

Since the implementation of Kerberos authentication, the use of a special *k5arc* account on a distribution host provides a technique for restricting access to the host. The account gets created when `ups installasroot k5arc` is run at installation time. The account's login shell is restricted to **`ksu`**.

The `k5arc` script (see section 21.3.1 *Scripts Used to Access Distribution Database*) is designed to take you to the *k5arc* account on a distribution host, then **`ksu`** you to the account of your choice, from where you can execute commands. The individual accounts on the distribution host contain `.k5users` files[1]. These files control which commands each user/principal is allowed to run.

21.3 Web Server Configuration

The Web server on the distribution node is used for two purposes:

- to run queries on the distribution node **UPS** database(s)
- to request that new products added to the server be filed away and declared

It may of course also be acting as a more general purpose Web server, however this makes the configuration somewhat more complex (the environment for execution of the cgi script needed for the distribution node activities may need to be different than the environment for the other activities of the server).

If it is only performing tasks related to distribution node activities, it is reasonable for the Web server to run directly as the *updadmin* account. Then all of its cgi and k5arc scripts, etc. will be run as that account. If, on the other hand, other unrelated tasks are being performed on the Web server, steps should be taken to ensure that the **UPD** cgi and k5arc scripts get executed as the *updadmin* account, while other activities are performed under whatever

1. Described in the Strong Authentication document at `http://www.fnal.gov/docs/strong-auth/html/user.html#28738`

account is appropriate for them. Configuring the Web server to handle other tasks as well as running the **UPS**/**UPD** cgi scripts is beyond the scope of this document.

We recommend you use the **apache** product for your Web server, which you can install using **`upd install apache`** and tailor with **`ups tailor apache`**. Tailor it such that it runs as the account *updadmin*, and the configuration files are owned by the administrative account *wwwadm*.

21.3.1 Scripts Used to Access Distribution Database

ups.cgi Used to Query the Database

Any **UPS/UPD**-related cgi scripts must reside in the Web server's `/cgi-bin` area. Currently there is only one: `ups.cgi`. This script gets called by some **UPD** commands, and determines if the product exists on the server, and what its dependencies are.

This script is provided in the **UPD** product itself, in `$UPD_DIR/cgi-bin`. We recommend that you make the Web server's version of this file a symbolic link to the `$UPD_DIR/cgi-bin` version.

Note that when new versions of **UPD** are installed on the server, the Web server's version of this file needs to be manually updated! This does not happen automatically because the **UPD** product doesn't know where the **apache** product has the `cgi-bin` areas for its respective products.

k5arc Scripts Used to Add/Modify Products

These scripts are maintained on the distribution host in the `/usr/krb5/k5arc/scripts` directory, which needs to be in the $PATH for the account *updadmin*.

`upd`	installs the product on the distribution node
`ups`	declares the product into the distribution database

When a user runs **`upd addproduct`** or **`upd modproduct`** from a client machine, the command calls the local `k5arc` script which then gets invoked behind-the-scenes with three arguments:

```
% k5arc <host> <user> <command>
```

Let's look at how the `upd` script would get run. Say the distribution host is fnkits. To install a product, the *updadmin* account must be used, and the command is the name of the script (`upd`), plus any arguments (arguments would be based on the **`upd addproduct`** command string invoked by the user):

```
% k5arc fnkits updadmin upd [options_inferred_from_command]
```

What is `k5arc` really doing? The script essentially provides shorthand for the following command:

```
% rsh -f -l k5arc <host> ksu -n $<you> <user> -e <command>
```

which runs Kerberized **`rsh`** to the specified distribution host while forwarding your Kerberos credentials, brings you into the *k5arc* account, runs **`ksu`** to a different account of your choice (**`<user>`**), and from there runs your specified command (or script). Note that **`-n ${you}`** reminds **`ksu`** of your Kerberos principal (e.g., **`ksu -n joe@FNAL.GOV`**); this gets filled in automatically as long as you're authenticated to the FNAL.GOV realm.

21.3.2 Restricting Access to Distribution Database

It is critical to maintain control over the distribution database. Restricting access is now done via a program that utilizes Kerberos authentication, **k5arc**. The `k5arc` script comes with **UPD** and gets installed in the client machine's `/usr/krb5/k5arc` area. It gets called by **`upd addproduct`** and **`upd modproduct`** to run scripts that add or modify products on a distribution server (see section 21.3.1 *Scripts Used to Access Distribution Database*). It can also be invoked to run a generic command.

We use `k5arc` to set permissions for individual Kerberos principals on the distribution host. To allow a user/principal to add and modify products, you as the distribution host administrator with root privileges can run (from a client machine):

```
% k5arc <host> root allow_addproduct <principal>@FNAL.GOV
```

where `allow_addproduct` is an admin script that simplifies what you otherwise would need to type, namely the two commands:

```
% k5arc <host> root admin allow <user>@FNAL.gov updadmin ups
```

```
% k5arc <host> root admin allow <user>@FNAL.gov updadmin upd
```

You can also give users limited admin rights, e.g., the right to give yet other users the `allow_addproduct` permission. For example, say you want the person whose principle is mary@FNAL.GOV to set users up to add and modify products on fnkits. You would enter the command:

```
% k5arc fnkits root admin allow mary@FNAL.GOV root
  allow_addproduct
```

Example

I'm the administrator of the system, and I've done the **`ups installasroot k5arc`** on fnkits, and therefore have root access and can run the **`admin`** command. I want to find out what principals have what privileges (initially I should see only my principal):

```
% k5arc fnkits root admin list
```

```
              Principal      User Command
              ---------      ---- -------
            me@FNAL.GOV      root admin
```

At this point I am the only one on the machine who can do anything with k5arc. Now I want hermione@FNAL.GOV to be able to give people permission to run **`upd addproduct`**.

First I put the `allow_addproduct` script in `/usr/krb5/k5arc/scripts`, then I run:

```
% k5arc fnkits root admin allow hermione@FNAL.GOV root
  allow_addproduct
```

Now **`k5arc fnkits root admin list`** yields:

```
              Principal      User Command
              ---------      ---- -------
            me@FNAL.GOV      root admin
      hermione@FNAL.GOV      root allow_addproduct
```

At this point Hermione can give ron@FNAL.GOV permission to add products by running:

```
% k5arc fnkits root allow_addproduct ron@FNAL.GOV
```

And now **`k5arc fnkits root admin list`** yields:

```
              Principal      User Command
              ---------      ---- -------
            me@FNAL.GOV      root admin
      hermione@FNAL.GOV      root allow_addproduct
           ron@FNAL.GOV  updadmin ups
           ron@FNAL.GOV  updadmin upd
```

Only Ron can do an addproduct, but I (me@FNAL.GOV) could set permissions to allow myself to do it.

21.3.3 Prerequisites for Modifying the Distribution Database

There are a few prerequisites in order for the Web server's cgi script to run:

- **UPD** must be setup (the **apache** product in `KITS` has its scripts configured to setup **UPS**, **perl** and **python** when launching a Web server).
- The variable $PRODUCTS must be set to the database list for product distribution.

UPS needs to be setup because the `ups.cgi` script performs **UPS** commands, and to do so, the script must be able to determine the database using the $PRODUCTS path. **UPD** must be setup so that the cgi scripts can find $UPD_DIR and the associated $UPD_USERCODE_DIR to find the **UPD** configuration.

To ensure that these things are done for the *fnkits* Web server, we have added the following lines to its (**apache**) `admin/start` script:

```
#-----------------------------------------------
# added for upd server, start upd, set products
umask 002
setup upd
PRODUCTS=/ftp/upsdb; export PRODUCTS
#-----------------------------------------------
```

21.3.4 Permissions on Files Created in the Distribution Database

Notice that the text in the start script for *fnkits* shown in section 21.3.3 *Prerequisites for Modifying the Distribution Database* sets the **umask** with which the cgi script will be run. This affects the permissions on all the files generated by any cgi script run by the Web server; in particular, files created in the distribution **UPS** database, product tar files and table files, and so on.

If a tighter **umask** than **`002`** is used, it tends to "turn off" permissions in **UPS** product directories, which are then not appropriately group-writable when installed on end user systems.

21.4 FTP Server Configuration

For the **FTP** server on your distribution node, we recommend the one in the **wu_ftpd** product, which is available from *fnkits*. This product expects to find its configuration files in `/etc/ftpd`. This directory needs to be made writable by your *wwwadm* account, or equivalent, and to be configured to grant

access to the same groups/individuals as for your Web server. We recommend that you configure the `/etc/ftpd/ftpaccess` file as shown (explanations follow the file listing):

```
class   local   real,anonymous  *.fnal.gov
class registeredhost anonymous registered.host.1
# ... lots more of these

# -------------------------------
limit   local 100   Any
log commands anonymous,real
log transfers anonymous,real inbound,outbound
chmod no anonymous
delete no anonymous
overwrite no anonymous
rename no anonymous
umask no anonymous

# anybody can do tar and compression
compress        yes      *
tar             yes      *

upload /ftp *           no
upload /ftp /incoming   yes updadmin upd 0640 nodirs

private yes

autogroup upd local
autogroup upd registeredhost

message /etc/welcome.msg        login
message  /etc/upd.msg                   login     local  gupd
registeredhost
message /etc/upd-was.mesg       login   wasregistered
message /etc/non-upd.msg        login   all

# ------------------------
class   remote  real,anonymous  *
```

This configuration accomplishes the following things:

- specifies several classes of users with `class` directives for `local`, `registered host`, and (at the end) `remote`; the `local` and `registeredhost` classes get mapped to the `upd` group in the `autogroup` lines.
- turns on logging with `log` directives
- restricts anonymous **FTP** users from doing anything dangerous, via `chmod delete`, `overwrite rename` and `umask` directives
- allows compression, and "tarring" of files with the `tar` and `compress` directives

- only allows `uploads` into `/incoming` under the `ftp` area (Note that this is redundant given the file permissions, but redundancy is sometimes good!)
- specifies different login messages with the `message` directive to let users coming in directly by **FTP** know what they're allowed to download.

Don't forget to recheck the account setup issues for the *ftp* account in section 21.2.2 *The ftp Account*.

The other **FTP** configuration files should be empty initially, except for `ftpconversions`; where the stock file from `$WU_FTPD_DIR/examples` should be sufficient. You can add entries to `ftpgroups` later to implement proprietary-style access control on some products, if needed.

The **FTP** configuration files should be writable by the *wwwadm* account.

21.5 UPD Configuration Items

There are several **UPD** configuration items in the `${UPD_USERCODE_DIR}/updconfig` file that are used exclusively on product distribution servers. These must be set properly in order to have a working install server.

21.5.1 Archive File Keywords and ${SUFFIX}

The special keywords are:

UNWIND_ARCHIVE_FILE	the absolute path to the archive file (not unwound)
UPS_ARCHIVE_FILE	the path that **UPD** uses to declare the archive file to **UPS** (minus `ftp://<host>` which gets prepended before it is declared[1])

UNWIND_ARCHIVE_FILE and UPS_ARCHIVE_FILE are similar to other **UPD** configuration file variable pairs (see section 32.3.1 *Required Locations*). The values for both these variables are paths that must end in the file name appended by ${SUFFIX}. ${SUFFIX} is a read-only variable that describes the type of archive (e.g., `tar`, `tar.gz`, `zip`). Its value comes from the **UPD** command line. Including ${SUFFIX} on the end of the definition is *mandatory*; **UPD** cannot install a product whose archive file does not end in the proper suffix.[2]

1. **`<host>`** is the current **`-h <host>`** argument to **`upd addproduct`**. It defaults to *fnkits.fnal.gov.*

Examples

Here are sample definitions of UNWIND_ARCHIVE_FILE and UPS_ARCHIVE_FILE:

```
UNWIND_ARCHIVE_FILE="/ftp/archives/${UPS_PROD_NAM
E}${UPS_PROD_VERSION}${UPS_PROD_FLAVOR}${UPS_PROD
_QUALIFIERS}.${SUFFIX}" (all on one line in real file)
UPS_ARCHIVE_FILE="${UNWIND_ARCHIVE_FILE}"
```

- `VE_FILE="${UNWIND_ARCHIVE_FILE}"`

UPD will then use the following path to declare the product tar file to **UPS**:

```
ftp://<host>/ftp/archives/${UPS_PROD_NAME}${UPS_
PROD_VERSION}${UPS_PROD_FLAVOR}${UPS_PROD_QUALIFI
ERS}.${SUFFIX}
```

To make **UPD** use an **FTP** server at a particular port number (e.g., 777), define UPS_ARCHIVE_FILE as:

```
UPS_ARCHIVE_FILE=":777${UNWIND_ARCHIVE_FILE}"
```

21.5.2 Pre- and Postdeclare ACTIONs

As with any `updconfig` file, you can define pre- and/or postdeclare actions. These are described in section 32.4 *Pre- and Postdeclare Actions*. Briefly, they define actions for **UPD** to take just before or just after declaring a product to the database. They can be used for a number of tasks on a distribution server, e.g.,

- to apply permission file changes
- to add symbolic links
- to update html index files

In particular, when combined with use of the **optionlist** product, described in section 21.6.5 *Flagging Special Category Products Using Optionlist*, you can cause certain products to use a stanza in the configuration file that sets special group access permissions on the files that have just been installed. For example, the following text shows a predeclare action that makes the files for some product on *fnkits* readable only by group FNALONLY, which in combination with the **FTP** server configuration, means that only users in the `.fnal.gov` domain can access those files:

```
action = predeclare
    #
    # fix group permissions
    #
```

2. You could in principle use a specific suffix, e.g., ".tar" in place of ${SUFFIX} if the actions in the file don't repack or compress/decompress the files.

```
            Execute("chgrp  FNALONLY  ${UNWIND_TABLE_DIR}/*",
NO_UPS_ENV)
               Execute("chmod  o-rwx  ${UNWIND_TABLE_DIR}/*",
NO_UPS_ENV)
           Execute("chmod  a+r  ${UNWIND_TABLE_DIR}/*.table",
NO_UPS_ENV)
```

21.6 Administrative Tasks and Utilities

21.6.1 Reporting FTP and Web Server Activity Using Ftpweblog

For reporting, we recommend the **ftpweblog** product, which allows you to build reports combining **FTP** and Web accesses. The **apache** product's tailor script writes a statistics script, `monthlystats`, that can be modified to include calls to **ftpweblog** to request the **FTP** transfers for a **UPD** server machine. Then, whenever `monthlystats` gets run, you'll generate a combined report in your Web server's log summary area.

To modify `monthlystats` to get this information, first change the list of logfiles (`loglist`) to include the **FTP** `xferlog` file. Change

```
loglist="$accesslog $errorlog $agentlog $referlog"
```

to

```
loglist="$accesslog    $errorlog    $agentlog    $referlog
/var/log/ftpd/xferlog"
```

Next, just after the call to **ftpweblog** that's already there, add an extra invocation of **ftpweblog** to include the `xferlog` data for "today" in the summary report, e.g.,

```
...
mv ${thismonth} ${thismonth}.bak
totals="${thismonth}.bak"
ftpweblog                    \
-N "${title}"                \
-i ${totals}                 \
-t ftp /var/log/ftpd/xferlog \
 > ${thismonth}
...
```

21.6.2 Restricting Access for Uploads to Distribution Database

See section 21.3.2 *Restricting Access to Distribution Database*.

21.6.3 Restricting Access for Downloads from Distribution Database

In order to register particular hosts for downloading products, you need to add their hostnames to access files for both the Web and the **FTP** servers, as described in sections 21.3.2 *Restricting Access to Distribution Database* and 21.6.4 *Restricting Distribution of Particular Products*, respectively. The **FTP** access file is maintained at `/etc/ftpd/ftpaccess`, and the Web access file is found relative to the Web server directory at `./conf/access.conf`.

On *fnkits* we use the **`cmd addkits`** script to add hosts to the appropriate files. This script can be found in the `$UPD_DIR/admin` directory.

21.6.4 Restricting Distribution of Particular Products

The best available mechanism for limiting distribution of particular products is via group ids, using the **FTP** server's ability to map particular classes of clients to particular groups. By making a set of files under `~ftp` readable by a particular group only, the **FTP** server automatically restricts access to those files, allowing access only to those clients which are mapped to that group. For example, on *fnkits.fnal.gov* there is a group for registered users, a separate group for each proprietary product, and a group for on-site-only access. Groups can be created as you need them.

Note that if you do create such groups, you must either include the *updadmin* account in each group so that it has permission to change files to these groups (via **chgrp**), or use a mechanism like **cmd** or **sudo** to allow the *updadmin* account to do this.

21.6.5 Flagging Special Category Products Using Optionlist

UPD supports creation and use of a special table-file-only product called **optionlist** on a **UPD** server. In this table file, you can define options specific to products which may subsequently get installed on the server[1]. **UPD** checks this table file automatically when executing **upd install**, **upd addproduct**, or **upd modproduct**. **optionlist** provides a check in case a product provider forgets to flag a product as belonging to a special category.

upd install only looks for "proprietary" in the options, to see if it should prompt the user for account information and do SITE GROUP commands, and so on. The **upd addproduct** and **upd modproduct** commands pass any listed option(s) over to the **upd move_table_file**, etc., commands on the server side, thereby setting the listed flags as if they had been put on the command line with the **-O** option. In other words, **UPD** effectively ignores all options except **proprietary**, and just passes them through to the **UPD** configuration on the server.

The **optionlist** product should be declared as version "only" and flavor "NULL". It requires user-defined keywords of the form `_UPD_OPTS_<PRODUCT>="<option_list>"` (see section 28.2 *Keywords: Information Storage Format*) defined in `updconfig`, and uses them as shown in the example below:

```
FILE=Table
Product=optionlist

group:
Flavor=ANY
Qualifiers=""

common:
# Proprietary products
_upd_opts_edt="proprietary"
_upd_opts_flint="proprietary"
end:
```

1. Really early versions of **UPD** used a **proprietarylist** product for proprietary products; the process has now been generalized to include other product types.

According to this table file, whenever an instance of **edt** or **flint** gets installed on the server, **`upd install`** gets run with the option **`-O proprietary`**. These products will only match a `updconfig` file stanza that specifies `options=proprietary`.

You can download the current *fnkits* **optionlist** table file for reference by issuing the command:

```
% upd fetch -J @table_file optionlist
```

This gets the file `optionlist_only_NULL.table`. There are about 80 entries in the file at the time of this writing.

21.6.6 Searching FTP Server Logfiles Using Searchlog

On *fnkits* we have a simple cgi script that lets users search for downloads/uploads of particular products in the **FTP** server logs. It can be run from `http://fnkits.fnal.gov/cgi-bin/searchlog.cgi`. The file content is shown here:

```
#!/bin/sh

# adapt the following to find your xferlogs if needed
#
logfiles="‘echo /var/adm/xferlog* /var/log/ftpd*/xferlog*‘"

echo "Content-type: text/html"
echo

if [ $# = 0 ]
then
    cat <<EOP
         <html> <head> <title> upd Downloads Search </title>
</head>
        <body> <h1> upd Downloads Search </h1>
        Please enter a product and version
        <isindex> </body> </html>
EOP

else
    if [ $# != 2 ]
    then
        cat <<EOP
       <html> <head> <title> Invalid Search </title> </head>
        <body> <h1> Inalid search </h1>
         Please use your back button and enter a product and
version!
        </body> </html>
```

```
EOP
    else
        cat <<EOP
       <html> <head> <title> Search Results </title> </head>
        <body> <h1> Search Results </h1>
        Search results for product $1 version $2
        <pre>
EOP
        for f in $logfiles
        do
          case $f in
          *.gz|*.Z) gunzip < $f ;;
          *) cat $f ;;
          esac
        done 2>/dev/null | grep "$1" | grep "$2"
        cat <<EOP
        </pre> </body> </html>
EOP
    fi
fi
```

21.7 Product Distribution via CD-ROM

A CD-ROM can be used as a distribution database. You start with a local directory tree containing the necessary files and products, and then, using appropriate tools, you create an image of this area and burn it onto a CD. The Computing Division has created images for use on Linux machines. You can obtain a CD-ROM of one of the images, use one of the images to create your own CD-ROM, or if none of the provided images meets your needs, you can create your own image and make a CD-ROM from that.

In `ftp://ftp.fnal.gov/products/bootstrap/current` there is a script called `mkcdrom`, which makes a file tree which can be put on a CDROM. It takes several options:

`-f <flavor1>:<flavor2>`:	...Put products/bootstraps for each flavor listed, default is output of **`ups flavor -2`**
`-b`	include **UPS/UPD** bootstrap sets in the CDROM area
`-h <host>`	use host (default fnkits.fnal.gov) to find products
`<directory>`	directory in which to put the files (required)
`<product>`	product whose dependencies to include in the area

Say, for example, you wanted to make a CDROM on Linux[1] of **CoreFUE**, **gcc** and **python** which someone could use to bootstrap **UPS/UPD**. Here's how you would do it (some of the following commands produce volumes of output):

```
% mkdir /tmp/bigdir                    # directory to use
% mkcdrom -b /tmp/bigdir coreFUE       # add some products
% mkcdrom /tmp/bigdir gcc              # add some more
% mkcdrom /tmp/bigdir python           # add some more
```

Before you proceed to the next two commands, read the document at `http://www.tldp.org/HOWTO/CD-Writing-HOWTO.html`, and follow the instructions there. Then run:

```
% mkisofs -LlNRv -o /tmp/bigdir.iso   # make cdrom image
% cdrecord dev=0,0,0 /tmp/bigdir.iso  # burn image on CD
```

To proceed to bootstrap **UPS/UPD** (**CoreFUE**) from this, you'd run (shown for **bootstrap** v2_1)[2]:

```
% mount /mnt/cdrom
% cd /mnt/cdrom/prd/bootstrap/v2_1/configs
% sh ../stage1.sh local
```

Then you'd need to setup **UPS/UPD** and install the other products (**gcc** and **python**):

```
% . /usr/local/etc/setups.sh
% setup upd
% upd install -h file://localhost/mnt/cdrom/db gcc
% upd install -h file://localhost/mnt/cdrom/db python
```

1. Finding out how to burn a CDROM on Solaris or IRIX is left as an exercise to the reader(!).
2. The `mount` command for CDROM for Solaris and/or IRIX and/or OSF1 requires a few extra options; check the man pages.